2. Benjamin

Program Name: Benjamin.java **Input File: benjamin.dat**

Benjamin has decided to expand the stack and queue process he just learned in class by adding some new processing features. Besides the normal push and pop, he adds low, high, and middle processes, either inserting a value or removing a value using these concepts. For working with stacks of digits, he decides on eight operations, which are as follows:

1. P - Pop a digit from the top of the stack.
2. P(X) - Push X onto the top of the stack.
3. L - Remove the lowest valued digit from the stack.
4. L(X) - Insert X immediately below the lowest valued digit in the stack.
5. H - Remove the greatest valued digit from the stack.
6. H(X) - Insert X immediately above the greatest valued digit in the stack.
7. M - Remove the digit at the middle* of the stack.
8. M(X) - Insert X so that it is positioned as the new middle* of the stack.

For queues of digits, he will use the same eight operations, as defined below:
1. P - Pop a digit from the front of the queue.
2. P(X) - Push X onto the back of the queue.
3. L - Remove the lowest valued digit from the queue.
4. L(X) - Insert X immediately in front of the lowest valued digit in the queue.
5. H - Remove the greatest valued digit from the queue.
6. H(X) - Insert X immediately behind the greatest valued digit in the queue.
7. M - Remove the digit at the middle* of the queue.
8. M(X) - Insert X so that it is positioned as the new middle* of the queue.

*NOTE: For a stack or queue with an even number of digits, Benjamin observes that the "middle" could be one of two different digits. He decides that, for his purposes, "middle" shall refer to the digit from the middlemost pair that is closer to the bottom of the stack or the front of the queue. For example, in the list of digits below, the "middle" digit is the 6 (not the 1), whether the digits are in a stack or a queue:

Bottom/Front --> [5, 3, 7, 6, 1, 9, 8, 7] <-- Top/Back

Input - An initial string of digits in the range 0-9, all on the first line, with single space separation. The values in this list are to be used to create both a stack and a queue by adding each digit in order, from left to right. On the next several lines will be commands that will either affect the stack or the queue, indicated by either the letter S or Q, followed by one or more commands to be evaluated left-to-right as defined above. When an L command is given, you may assume that the stack or queue contains only 1 digit that is strictly less than all other digits. Similarly, you may assume that when an H command is given, the stack or queue contains only 1 digit that is strictly greater than all others. No commands will result in attempting to remove a digit from an empty stack or queue.

Output - After processing each line of commands, print the modified stack or queue, with the corresponding letter (S or Q) followed by its list of contents, as shown below. Stacks should be printed with the bottom of the stack to the left and the top of the stack to the right. Queues should be printed with the front of the queue to the left and the back of the queue to the right.

Sample data:

```
5 3 7 6 1 9 8 7
S P
Q P(4)
Q L M(2)
S L(0) H(2)
S L M
```

Sample Output:

```
S [5, 3, 7, 6, 1, 9, 8]
Q [5, 3, 7, 6, 1, 9, 8, 7, 4]
Q [5, 3, 7, 6, 2, 9, 8, 7, 4]
S [5, 3, 7, 6, 0, 1, 9, 2, 8]
S [5, 3, 7, 1, 9, 2, 8]
```